Remarks

Claims 1-17 are pending in this application. Claims 1-17 have been rejected. No claims have been amended.

I. Nonstatutory Double Patenting Rejection of Claims 1-17 Over Co-pending Patent Application Serial No. 09/706050

Claims 1-17 stand provisionally rejected under the judicially-created doctrine of double patenting over claims 1-10 of copending Patent Application Serial No. 09/706050. In response, a Terminal Disclaimer is being filed with this Response to overcome the provisional rejection for double patenting. Applicant therefore respectfully requests that the provisional double patenting rejection of claims 1-17 be withdrawn.

II. The Rejection Under 35 U.S.C. § 103(a) of Claims 1-17 Over You is Improper and Should be Withdrawn

Claims 1-17 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over You (U.S. Patent No. 6,158,045), in light of "Compilers: Principles, Techniques, and Tools," by Alfred V. Aho, Ravi Stehi, and Jeffrey D. Ullman, pages 432-433, 439, and 703-711, and further in view of U.S. Patent Application Pub. No. 2003/0200397 to McAllister et al.

As the Office Action acknowledges, neither You nor Aho et al. discloses a step of gathering, extracting, or collecting "data . . . while maintaining data coherency," as required by all of claims 1-17, as presently in the application. The Office Action states, however, that McAllister discloses memory transaction coherency through the use of a memory controller agent. In particular, the Office Action states that McAllister discloses an "agent [which] is responsible for ensuring coherency and fulfilling memory transactions for a single memory line, thereby simplifying the design of the agent."

The Office Action fails, however, to point to any teaching, suggestion, or motivation in the above-cited references to combine the teachings of the references with each other to obtain the presently-claimed invention. For example, according to You:

> Traditional program debuggers are limited in many aspects. Some debuggers are dedicated to debug programs generated by a particular development environment. Others may be semiportable but may not be able to target more than one environment at a time. Still others can only debut single process programs, and are not truly interactive. (Col. 3, lines 42-46.)

You therefore states that "it is an object of the present invention to provide a portable and dynamic process for the debugging of computer programs which promotes better developer focus and concentration, and hence greater productivity" (col. 4, lines 37-40). In particular, You discloses "an architecture for debugging services" referred to as the "portable debugging services (PDS) architecture" (col. 9, lines 16-28). According to You, PDS "is portable in that a single architecture and single framework can be executed on a wide variety of platforms, which may vary in their microprocessor family, number of processors, processor register sizes, numbers of processor registers, memory addressing, operating system, . . . and a multitude of other parameters" (col. 9, lines 17-25).

You further states that:

> The portable debugging services (PDS) architecture is composed of a client-server program model. In this model, a single debugger server executes on the target host. A host-specific implementation which uses the PDS framework uses features in the host operating system for determining debugging information. (Col. 9, lines 16-32.)

FIG. 2 of You "illustrates a client-server debugging system in accordance with a preferred embodiment" (col. 9, lines 40-41). According to You, "[t]he client and server use many abstractions to encapsulate various information about process state, thread state,

execution state, and so on" (col. 10, lines 4-7). A substantial portion of the disclosure of You is devoted to describing the use of software objects, defined in accordance with an object-oriented programming language such as C++, to exchange debugging information between the client and server (see, for example, the section entitled "Client-Server Abstractions," col. 10, line 19 - column 54, line 58). For example, You describes "primitive objects" as examples of such software objects:

> The primitive objects containing the data that are exchanged between the client and server encapsulate a wide range of information which describe the state of the target host and server, the target processes, the target threads, runtime information, breakpoint and watchpoint descriptions, client identification, stacks and register values, and software and hardware exception and handling parameters. (Col. 10, lines 20-27.)

In summary, You discloses the use of software objects to abstractly encapsulate various kinds of debugging information.

The invention of McAllister, in contrast, "is a memory controller that provides memory line caching and memory transaction coherency by using at least one memory controller agent" (paragraph 34). Although McAllister does use the term "coherency," McAllister uses this term in the context of the need of a "modern memory controller . . . to efficiently handle memory transactions from each processor and I/O unit [in a multi-processor and multi-I/O unit system], while keeping all cache memories coherent and arbitrating between separate memory transactions to the same memory line" (paragraph 3).

As described by McAllister, a multi-processor computer system "has a potentially large number of processors and I/O units, with each processor and I/O unit having one or more cache memories" (paragraph 7). As further described by McAllister, it is possible for different caches to contain different versions of data. McAllister defines a "cache coherency mechanism" as "[a] mechanism

that ensures that the updated version of data is utilized" when an access to memory is made in such a multi-processor system (paragraph 8). McAllister discloses one such cache coherency mechanism in the form of a "memory controller that provides memory line caching and memory transaction coherency by using at least one memory controller agent" (paragraph 34). In particular, in the disclosure of McAllister, "[e]ach memory controller agent has a memory line memory controller and a memory line coherency controller, along with a cache memory capable of caching the contents of a memory line along with coherency information for the memory line" (paragraph 34). McAllister, in summary, discloses a memory controller for providing coherency of data among cache lines of multiple caches in a multi-processor computer system.

The above-referenced Office Action fails to point to any teaching, suggestion, or motivation in You or McAllister for combining the subject matter disclosed in You and McAllister. In fact, neither You nor McAllister provides any such teaching, suggestion, or other motivation to combine one of the references with each other.

As described above, You and McAllister are related to different fields, address different problems, and provide unrelated solutions. For example, You discloses a software-based system for exchanging debugging-related information using abstract software objects to "promote better developer focus and concentration, and hence greater productivity", while McAllister discloses a memory controller for controlling cache memories in a multi-processor computer system. The Office Action fails to point to any motivation provided by the references for combining these teachings of You and McAllister.

Although the Office Action states that "it would have been obvious . . . to include details on data coherency when retrieving data from memory because fresh data, not stale, 'dirty' data is necessary when attempting to debug memory," the Office Action points to no evidence for such motivation in the cited references.

Instead, such motivation is impermissibly drawn using hindsight based on the teachings of the present application.

Furthermore, the Office Action fails to point out how to modify the teachings of You and/or Aho et al. based on the asserted teaching of McAllister to obtain the presently-claimed invention. In particular, the Office Action fails to point out how the software-based debugging system of You could be combined with the memory controller of McAllister, much less how they could be combined to obtain the claimed invention.

For at least the reasons stated above, the rejection of claims 1-17 fails to make out a *prima facie* case of obviousness under 35 U.S.C. § 103(a). Applicant therefore respectfully traverses this rejection and requests that it be withdrawn.

III. Conclusions

Any dependent claims of the present application which are not specifically mentioned above include all of the limitations of the above-referenced independent claims, and therefore patentably distinguish over the cited references for at least the same reasons.

In view of the foregoing amendments and remarks, this application should now be in condition for allowance. A notice to this effect is respectfully requested. If the Examiner believes, after this response, that the application is not in condition for allowance, the Examiner is requested to call the applicant's attorney at the phone number listed below.

If this response is not considered timely filed and if a request for extension of time is otherwise absent, applicant hereby requests any extension of time. Please charge any fees, or make any credits, to Deposit Account No. 08-2025.

Respectfully submitted,

2/13/2004

Robert Plotkin, Esq. Date

Reg. No. 43,861

Robert Plotkin, P.C.
45 Butternut Circle
Concord, MA 01742-1937
Ph: (978) 318-9914
Email: rplotkin@rplotkin.com